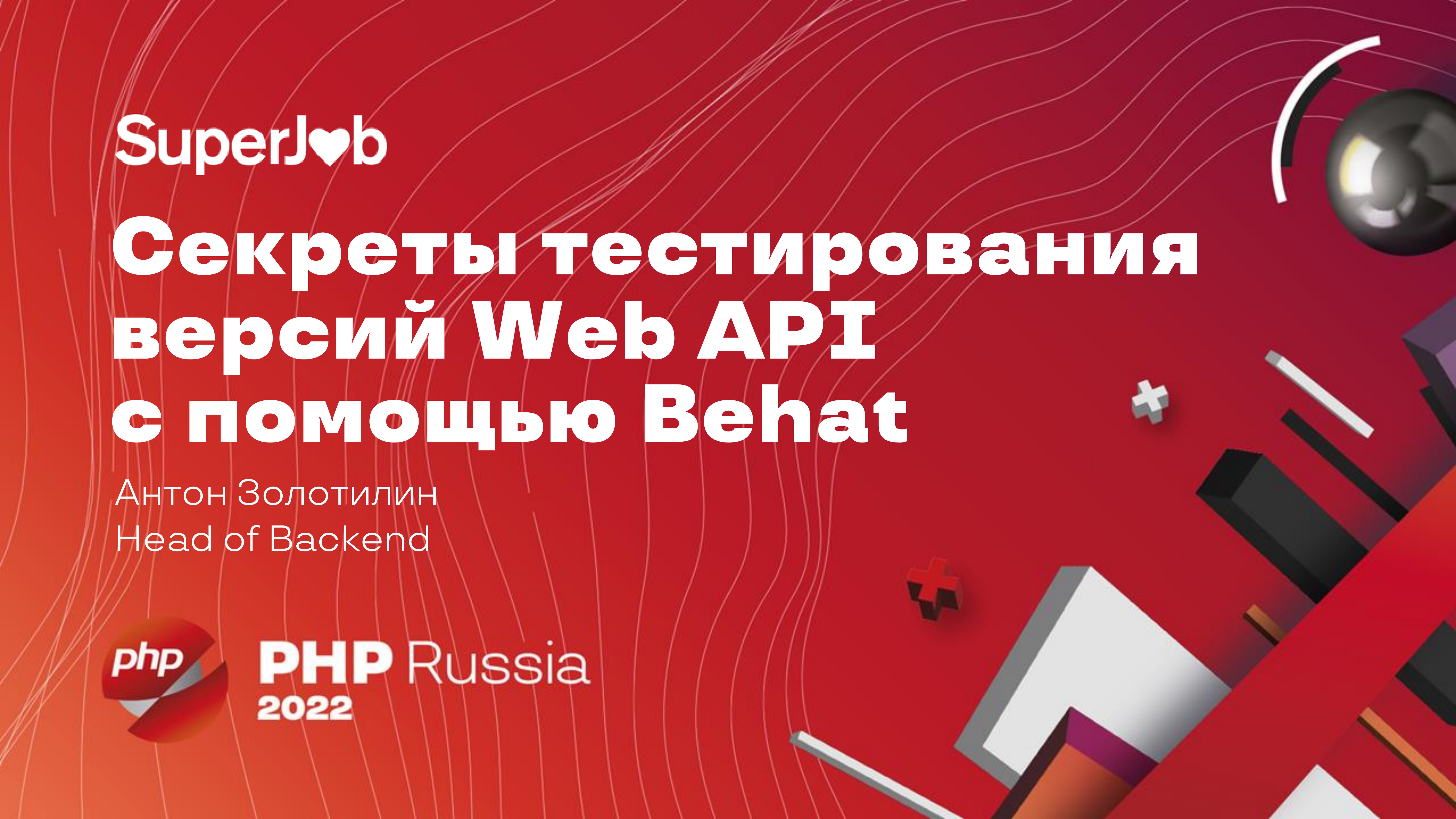

SuperJob
Секреты тестирования версий Web API с помощью Behat
Антон Золотилин
Head of Backend
php
PHP Russia
2022

Что такое SuperJob?

40 млн
резюме в базе

1,5 млн
пользователей в день

1 млн
вакансий в год





SuperJob API
и функциональное тестирование

18

поддерживаемых
мажорных версий

500

эндпоинтов в
последней версии

3 293*

тестовых сценариев

* уникальных





План

- Что такое функциональное тестирование, Behat и Gherkin?





План

- Что такое функциональное тестирование, Behat и Gherkin?
- Как работать с тестами в Behat и как их улучшить?





План

- Что такое функциональное тестирование, Behat и Gherkin?
- Как работать с тестами в Behat и как их улучшить?
- Как тестировать WebAPI, учитывая версии?





План

- Что такое функциональное тестирование, Behat и Gherkin?
- Как работать с тестами в Behat и как их улучшить?
- Как тестировать WebAPI, учитывая версии?
- Как мы доработали Behat для ускорения прохождения тестов?





Что такое Behat и с чем его едят

Если нечто выглядит как утка, плавает как утка и крякает как утка, то это, скорее всего, и есть утка





Немного определений





Немного определений

Функциональное тестирование — тестирование ПО в целях проверки реализуемости функциональных требований.





Немного определений

Функциональное тестирование — тестирование ПО в целях проверки реализуемости функциональных требований.

Behat — PHP-фреймворк с открытым исходным кодом, использующий синтаксис языка Gherkin и позволяющий осуществлять разработку согласно принципам BDD.





Немного определений

Функциональное тестирование — тестирование ПО в целях проверки реализуемости функциональных требований.

Behat — PHP-фреймворк с открытым исходным кодом, использующий синтаксис языка Gherkin и позволяющий осуществлять разработку согласно принципам BDD.

Gherkin — человеко-читаемый язык описания желаемого поведения системы.





Синтаксис Gherkin. Фичи и сценарии

```
Feature: Лаконичное описание тестируемого функционала фичи
    В определенных ролях мы будем совершать разные действия
    Для того, чтобы получить определенный результат
    Все равно эти строки не будут выполняться, это просто
    описание

Scenario: Какой-то use case, который надо протестировать
      Given пользователь с какой-то ролью
      And также имеющий такое-то состояние
      When предпринимает какие-то действия
      And дополнительно делает кое-что еще
      Then получает следующий результат, который мы проверяем
      And происходит что-то еще, что мы тоже хотим проверить
```





Синтаксис Gherkin. Фичи и сценарии

```
Feature: Лаконичное описание тестируемого функционала фичи
    В определенных ролях мы будем совершать разные действия
    Для того, чтобы получить определенный результат
    Все равно эти строки не будут выполняться, это просто
    описание

Scenario: Какой-то use case, который надо протестировать
     Given пользователь с какой-то ролью
     And также имеющий такое-то состояние
     When предпринимает какие-то действия
     And дополнительно делает кое-что еще
     Then получает следующий результат, который мы проверяем
     And происходит что-то еще, что мы тоже хотим проверить
```





Синтаксис Gherkin. Фичи и сценарии

```
Feature: Лаконичное описание тестируемого функционала фичи
    В определенных ролях мы будем совершать разные действия
    Для того, чтобы получить определенный результат
    Все равно эти строки не будут выполняться, это просто
    описание

Scenario: Какой-то use case, который надо протестировать
      Given пользователь с какой-то ролью
      And также имеющий такое-то состояние
      When предпринимает какие-то действия
      And дополнительно делает кое-что еще
      Then получает следующий результат, который мы проверяем
      And происходит что-то еще, что мы тоже хотим проверить
```





Синтаксис Gherkin. Фичи и сценарии

```
Feature: Лаконичное описание тестируемого функционала фичи
    В определенных ролях мы будем совершать разные действия
    Для того, чтобы получить определенный результат
    Все равно эти строки не будут выполняться, это просто
    описание

Scenario: Какой-то use case, который надо протестировать
      Given пользователь с какой-то ролью
      And также имеющий такое-то состояние
      When предпринимает какие-то действия
      And дополнительно делает кое-что еще
      Then получает следующий результат, который мы проверяем
      And происходит что-то еще, что мы тоже хотим проверить
```





Синтаксис Gherkin. Фичи и сценарии

```
Feature: Лаконичное описание тестируемого функционала фичи
    В определенных ролях мы будем совершать разные действия
    Для того, чтобы получить определенный результат
    Все равно эти строки не будут выполняться, это просто
    описание

Scenario: Какой-то use case, который надо протестировать
      Given пользователь с какой-то ролью
      And также имеющий такое-то состояние
      When предпринимает какие-то действия
      And дополнительно делает кое-что еще
      Then получает следующий результат, который мы проверяем
      And происходит что-то еще, что мы тоже хотим проверить
```





Синтаксис Gherkin. Фичи и сценарии

```
Feature: Лаконичное описание тестируемого функционала фичи
    В определенных ролях мы будем совершать разные действия
    Для того, чтобы получить определенный результат
    Все равно эти строки не будут выполняться, это просто
    описание

Scenario: Какой-то use case, который надо протестировать
      Given пользователь с какой-то ролью
      And также имеющий такое-то состояние
      When предпринимает какие-то действия
      And дополнительно делает кое-что еще
      Then получает следующий результат, который мы проверяем
      And происходит что-то еще, что мы тоже хотим проверить
```





Синтаксис Gherkin. Фичи и сценарии

```
Feature: Лаконичное описание тестируемого функционала фичи
    В определенных ролях мы будем совершать разные действия
    Для того, чтобы получить определенный результат
    Все равно эти строки не будут выполняться, это просто
    описание

Scenario: Какой-то use case, который надо протестировать
      Given пользователь с какой-то ролью
      And также имеющий такое-то состояние
      When предпринимает какие-то действия
      And дополнительно делает кое-что еще
      Then получает следующий результат, который мы проверяем
      And происходит что-то еще, что мы тоже хотим проверить
```





Предыстория (Background)

```
Feature: функционал доступности личного кабинета для пользователя

Background:
      Given пользователь с логином Vasya
      And пользователь имеет электронный адрес vasya@mail.ru
      And у пользователя нет ролей

Scenario: Вася пытается выдать себе админские права
      Given пользователь вошел в систему как Vasya
      When пользователь устанавливает себе роль Admin
      Then получает сообщение “Vasya, ты не прав, у тебя нет прав”
      And в логе появляется запись “Vasya пытался прогнуть систему”
```





Предыстория (Background)

```
Feature: функционал доступности личного кабинета для пользователя

Background:
      Given пользователь с логином Vasya
      And пользователь имеет электронный адрес vasya@mail.ru
      And у пользователя нет ролей

Scenario: Вася пытается выдать себе админские права
      Given пользователь вошел в систему как Vasya
      When пользователь устанавливает себе роль Admin
      Then получает сообщение “Vasya, ты не прав, у тебя нет прав”
      And в логе появляется запись “Vasya пытался прогнуть систему”
```





Шаблон сценария (Scenario Outline)

```
Scenario Outline: Шаблон сценария для теста элементарного сложения
      Given первое слагаемое равно X
      When прибавляем к нему Y
      Then должны получить в результате Сумма

Examples:
      |     X     |     Y     |      Сумма     |
      |     1     |     1     |     2     |
      |     10    |     0     |     10    |
      |     7     |    -7     |     0     |
```





Реализация Behat-тестирования

Бехаты — это просто. Надо взять обычный...

```
Feature: Функции калькулятора

Scenario: Тестируем сложение
      Given первое число равно 1
      When второе число равно 1
      Then сумма двух чисел дает в результате 2
```





Файл FeatureContext.php — реализуем шаги

```php
class FeatureContext extends SnippetAcceptingContext
{
      /**
      * @Given первое число равно :first
      * @Given /^первое число равно (\d+)$/
      */
      public function setFirst($first): void
      {
            $this->firstNumber = $first;
      }
}
```





Файл FeatureContext.php — реализуем шаги

```
class FeatureContext extends SnippetAcceptingContext
{
      /**
      * @When второе число равно :second
      * @When /^второе число равно (\d+)$/
      */
      public function setSecond($second): void
      {
            $this->secondNumber = $second;
      }
}
```





Файл FeatureContext.php — реализуем шаги

```
class FeatureContext extends SnippetAcceptingContext
{
    /**
    * @Then сумма двух чисел дает в результате :result
    */
    public function assertSumResult($result): void
    {
        Assert::assertEquals(
            $result,
            $this->calculator->getSum(
                $this->firstNumber,
                $this->secondNumber
            )
        );
    }
}
```





Улучшаем работу с тестами





Используйте плейсхолдеры

```
Feature: Функции калькулятора

Scenario: Тестируем сложение
  Given первое число равно 1
  When второе число равно 1
  Then сумма двух чисел дает в результате 2
```

```
class FeatureContext …
{

     public function setFirst($first): void
     {
        $this->firstNumber = $first;
     }

     public function setSecond($second): void
     {
        $this->secondNumber = $second;
     }
}
```





Используйте плейсхолдеры

```
Feature: Функции калькулятора

Scenario: Тестируем сложение
  Given первое число равно 1
  When второе число равно 1
  Then сумма двух чисел дает в результате 2
```

```
class FeatureContext …
{
     /**
     * @When :placeholder число равно :number
     */
     public function setNumber(
          $placeholder,
          $number
     ): void
     {
       $this
               ->placeholders
               ->setValue(
                    $placeholder,
                    $number
               );
     }
}
```





Используйте плейсхолдеры

```
Feature: Функции калькулятора

Scenario: Тестируем сложение
  Given первое число равно 1
  When второе число равно 1
  Then сумма #первое# и #второе# равно 2
```

```
class FeatureContext …
{
   /**
   * @Then сумма :first и :second равно :result
   */
   public function assertSum(
      $first,
      $second,
      $result
   ): void {
      Assert::assertEquals(
         $result,
         $this->calculator->getSum(
            $this->placeholders->replace($first),
            $this->placeholders->replace($second)
         )
      );
   }
}
```





Плейсхолдеры и контекст

```
Feature: Пользователи

Scenario: Устанавливаем связь через номер телефона
  Given I logged as new user
  And I’ve generated new random “phone”
  And set user phone “#random.phone#”
  And I’ve copied data “#currentUser#” to “friend”
  And I logged as new user
  When I set relationship by phone “#random.phone#”
  Then user has friend with id “#friend.id#”
```





Плейсхолдеры и контекст

```
Feature: Пользователи

Scenario: Устанавливаем связь через номер телефона
  Given I logged as new user
  And I've generated new random "phone"
  And set user phone "#random.phone#"
  And I've copied data "#currentUser#" to "friend"
  And I logged as new user
  When I set relationship by phone "#random.phone#"
  Then user has friend with id "#friend.id#"
```

```php
class FeatureContext …
{
	public function createUser(array $params = [])
	{
		return $this->tapi->createUser($params);
	}

	public function loginUser($user)
	{
		$this->placeholders->set(
			//'currentUser'
			self::PLACEHOLDER_CURRENT_USER,
			$this->tapi->loginUser($user)
		);
	}

	/**
	* @Given I logged as new user
	*/
	public function iLoggedAsNewUser()
	{
		$this->loginUser($this->createUser());
	}
}
```





Плейсхолдеры и контекст

```
Feature: Пользователи

Scenario: Устанавливаем связь через номер телефона
  Given I logged as new user
  And I’ve generated new random “phone”
  And set user phone “#random.phone#”
  And I’ve copied data “#currentUser#”  to “friend”
  And I logged as new user
  When I set relationship by phone “#random.phone#”
  Then user has friend with id “#friend.id#”
```

```
class FeatureContext …
{
    /**
    * @Given I’ve generated new random :type
    */
    public function iGenerateRandom($type)
    {
        $this->placeholders->set(
            "random.$type",
            $this->faker->generate($type)
        );
    }

    /**
    * @Given set user phone :phone
    */
    public function setCurrentUserPhone(string  $phone)
    {
        $phone = $this->placeholders->replace($phone);
        $this->tapi->setUserPhone($phone);
        //...обновляем текущего пользователя
    }
}
```



Плейсхолдеры и контекст

```
Feature: Пользователи

Scenario: Устанавливаем связь через номер телефона
  Given I logged as new user
  And I've generated new random "phone"
  And set user phone "#random.phone#"
  And I've copied data "#currentUser#" to "friend"
  And I logged as new user
  When I set relationship by phone "#random.phone#"
  Then user has friend with id "#friend.id#"
```

```
class FeatureContext …
{
    /**
    * @Given I've copied data :data to :custom
    */
    public function saveCustomDataToPlaceholder(
        $data,
        $custom
    ) {
        $data = $this->placeholders
            ->replace($data);
        $this->placeholders->set(
                $custom,
                $data
          );
     }
}
```





Плейсхолдеры и контекст

```
Feature: Пользователи

Scenario: Устанавливаем связь через номер телефона
  Given I logged as new user
  And I’ve generated new random “phone”
  And set user phone “#random.phone#”
  And I’ve copied data “#currentUser#” to “friend”
  And I logged as new user
  When I set relationship by phone “#random.phone#”
  Then user has friend with id “#friend.id#”
```





Тэги && хуки





Тэги && хуки

```
class FeatureContext …
{
	/**
	* @AfterScenario
	*/
	public function clearPlaceholders()
	{
		$this->placeholders->clear();
	}

	/**
	* @AfterScenario @users
	*/
	public function logoutUser()
	{
		$this->tapi->logout();
	}
}
```





Тэги && хуки

```
@users
Feature: Пользователи

@relationships
Scenario: Устанавливаем связь через номер телефона
  Given I logged as new user
  And I’ve generated new random “phone”
  And set user phone “#random.phone#”
  And I’ve copied data “#currentUser#” to “friend”
  And I logged as new user
  When I set relationship by phone “#random.phone#”
  Then user has friend with id “#friend.id#”
```

```
class FeatureContext …
{
      /**
      * @AfterScenario
      */
      public function clearPlaceholders()
      {
            $this->placeholders->clear();
      }

      /**
      * @AfterScenario @users
      */
      public function logoutUser()
      {
            $this->tapi->logout();
      }
}
```





Тестируем версионированное API







Тэги версий, плейсхолдер версии

```
Feature: Пользователи

@version-from-3.0 @version-to-8.0
Scenario: Изменение имени юзера
  Given  I logged as new user
  And I have payload from yaml:
  """
  profile:
    attributes:
      id: #currentUser.id#
      firstname: Моё Имя
      lastname: Фамилия
  """
  When I send entity via "PATCH" to "/#v#/profile/#currentUser.id#/" and get itself
  Then the response status is 200
  And the response value "firstname" is "Моё Имя"
  And the response value of "lastname" is "Фамилия"
```





Расширение Behat, iterate by version

Feature

GherkinParser

FeatureTester

ScenarioTester/
OutlineTester

BackgroundTester

StepTester

SuperJob



Расширение Behat, iterate by version

Feature

GherkinParser

FeatureTester

ScenarioTester/
OutlineTester

BackgroundTester

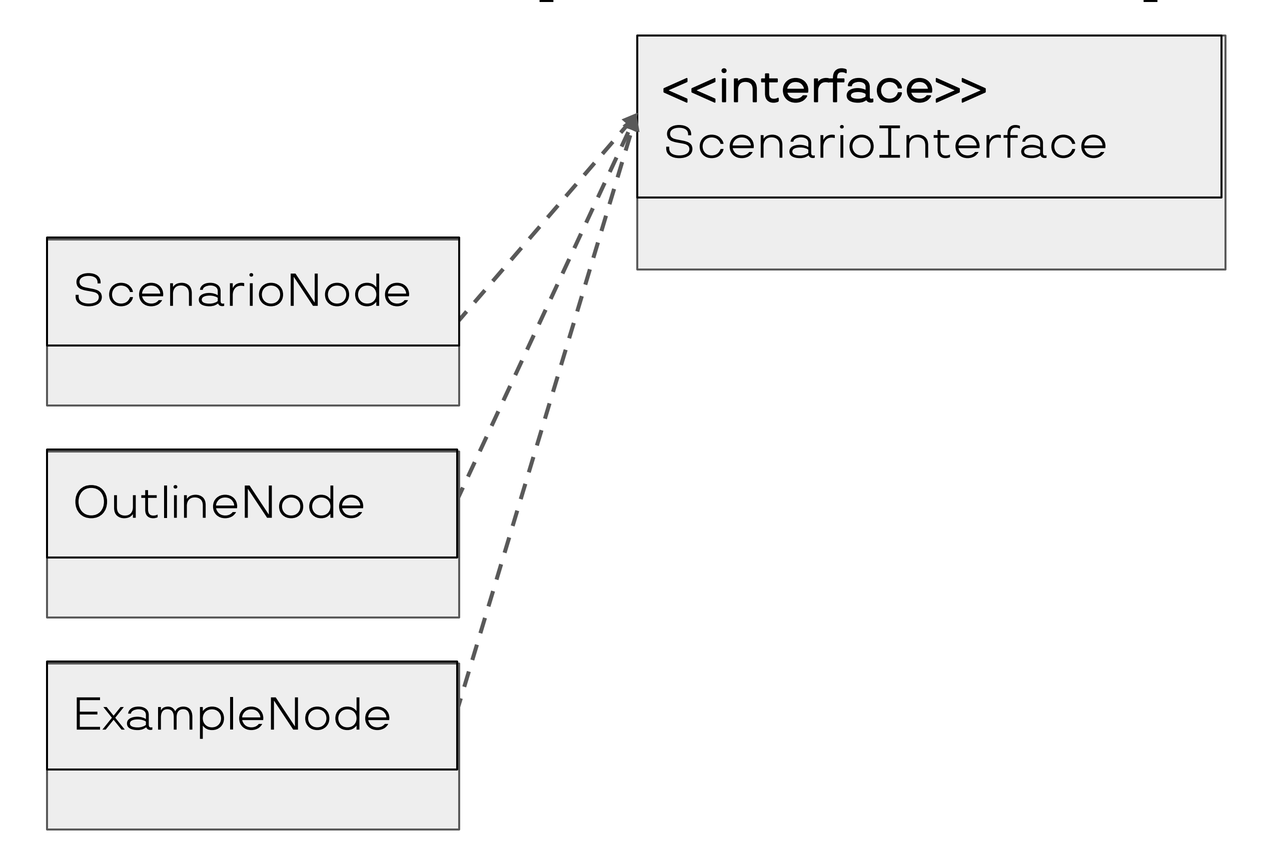
StepTester

From version A
To version B
iterate

SuperJob



Расширяем сценарий







Расширяем сценарий

<<interface>>
ScenarioInterface

ScenarioNode

OutlineNode

ExampleNode

VersioningScenario

+ version
+ originScenario

```
class VersioningScenario extends ScenarioNode
{
    private string $version;

    private ScenarioInterface $originScenario;

        ...
}
```

SuperJob



Подменяем IsolatingScenarioTester

IsolationScenarioTester

RuntimeScenarioTester

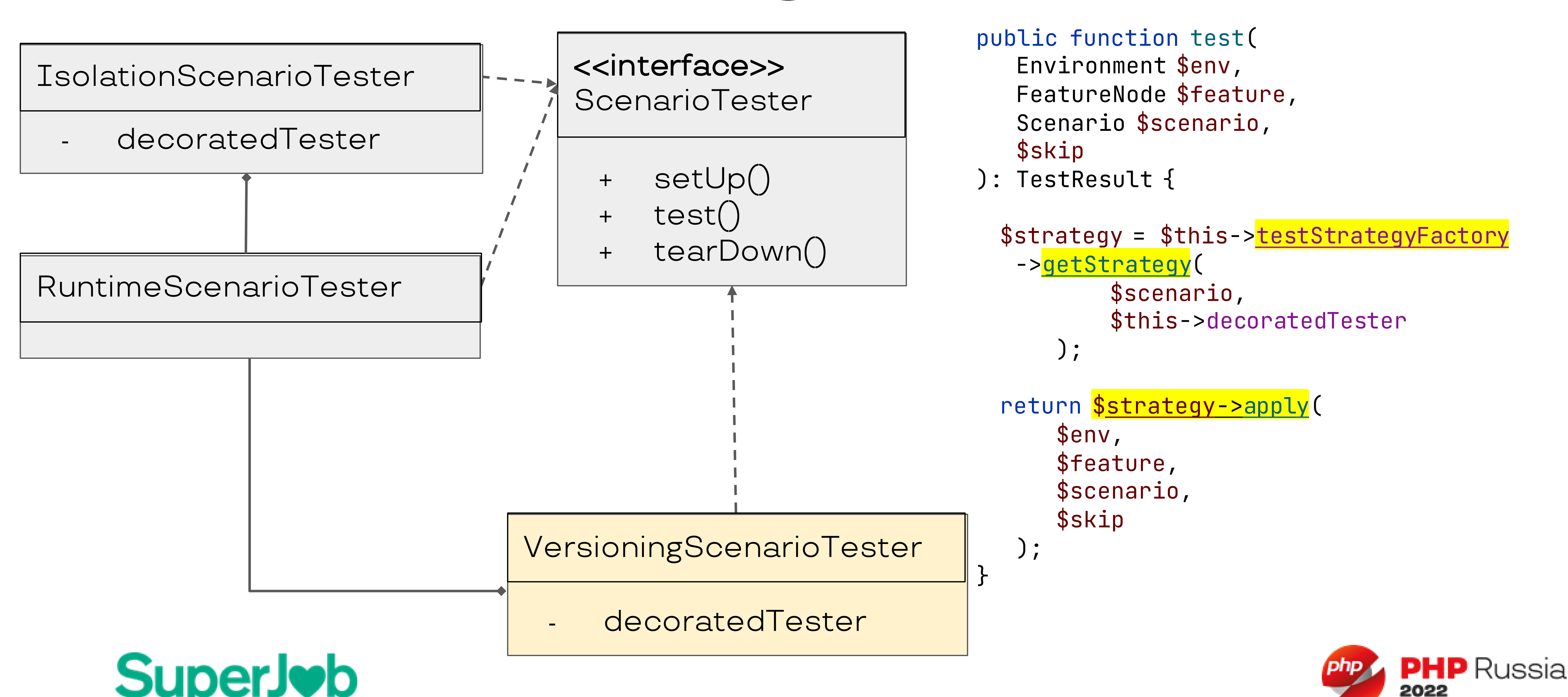
<<interface>>
ScenarioTester

+ setUp()
+ test()
+ tearDown()





Подменяем IsolatingScenarioTester



Фабрика стратегий тестирования

```
class StrategyFactory
{
    public function getStrategy(..., ScenarioInteface $scenario, ScenarioTester $decoratedTester)
    {
        $versionRange = $this->extractVersionRange($scenario);
        $strategyDigest = $versionRange->getDigest() . spl_object_hash($decoratedTester);
        return $this->strategies[$strategyDigest]
                ?? $this->createStrategy($versionRange, ...);
    }

    public function createStrategy(VersionRange $versionRange, ..)
    {
        $versionFilters = $this->versionFiltersFactory->getFilters($versionRange);
        $versionCollection = $this->tapi->getVersionCollection()->filter($versionFilters);
        return $this->strategies[$strategyDigest] = new VersionStrategy($versionCollection, ...);
    }
}
```





Фабрика стратегий тестирования

```
class StrategyFactory
{
   public function getStrategy(..., ScenarioInteface $scenario, ScenarioTester $decoratedTester)
   {
       $versionRange = $this->extractVersionRange($scenario);
       $strategyDigest = $versionRange->getDigest() . spl_object_hash($decoratedTester);
       return $this->strategies[$strategyDigest]
                ?? $this->createStrategy($versionRange, ...);
   }

   public function createStrategy(VersionRange $versionRange, ..)
   {
       $versionFilters = $this->versionFiltersFactory->getFilters($versionRange);
       $versionCollection = $this->tapi->getVersionCollection()->filter($versionFilters);
       return $this->strategies[$strategyDigest] = new VersionStrategy($versionCollection, ...);
   }
}
```





Фабрика стратегий тестирования

```
class StrategyFactory
{
   public function getStrategy(..., ScenarioInteface $scenario, ScenarioTester $decoratedTester)
   {
       $versionRange = $this->extractVersionRange($scenario);
       $strategyDigest = $versionRange->getDigest() . spl_object_hash($decoratedTester);
       return $this->strategies[$strategyDigest]
                 ?? $this->createStrategy($versionRange, ...);
   }

   public function createStrategy(VersionRange $versionRange, ..)
   {
       $versionFilters = $this->versionFiltersFactory->getFilters($versionRange);
       $versionCollection = $this->tapi->getVersionCollection()->filter($versionFilters);
       return $this->strategies[$strategyDigest] = new VersionStrategy($versionCollection, ...);
   }
}
```





Фабрика стратегий тестирования

```
class StrategyFactory
{
   public function getStrategy(..., ScenarioInteface $scenario, ScenarioTester $decoratedTester)
   {
       $versionRange = $this->extractVersionRange($scenario);
       $strategyDigest = $versionRange->getDigest() . spl_object_hash($decoratedTester);
       return $this->strategies[$strategyDigest]
                ?? $this->createStrategy($versionRange, ...);
   }

   public function createStrategy(VersionRange $versionRange, ..)
   {
       $versionFilters = $this->versionFiltersFactory->getFilters($versionRange);
       $versionCollection = $this->tapi->getVersionCollection()->filter($versionFilters);
       return $this->strategies[$strategyDigest] = new VersionStrategy($versionCollection, ...);
   }
}
```





Фабрика стратегий тестирования

```
class StrategyFactory
{
   public function getStrategy(..., ScenarioInteface $scenario, ScenarioTester $decoratedTester)
   {
       $versionRange = $this->extractVersionRange($scenario);
       $strategyDigest = $versionRange->getDigest() . spl_object_hash($decoratedTester);
       return $this->strategies[$strategyDigest]
               ?? $this->createStrategy($versionRange, ...);
   }

   public function createStrategy(VersionRange $versionRange, ..)
   {
   $versionFilters = $this->versionFiltersFactory->getFilters($versionRange);
       $versionCollection = $this->tapi->getVersionCollection()->filter($versionFilters);
       return $this->strategies[$strategyDigest] = new VersionStrategy($versionCollection, ...);
   }
}
```





Фабрика стратегий тестирования

```
class StrategyFactory
{
   public function getStrategy(..., ScenarioInteface $scenario, ScenarioTester $decoratedTester)
   {
       $versionRange = $this->extractVersionRange($scenario);
       $strategyDigest = $versionRange->getDigest() . spl_object_hash($decoratedTester);
       return $this->strategies[$strategyDigest]
                ?? $this->createStrategy($versionRange, ...);
   }

   public function createStrategy(VersionRange $versionRange, ..)
   {
       $versionFilters = $this->versionFiltersFactory->getFilters($versionRange);
       $versionCollection = $this->tapi->getVersionCollection()->filter($versionFilters);
       return $this->strategies[$strategyDigest] = new VersionStrategy($versionCollection, ...);
   }
}
```





Стратегия тестирования

```
class VersionStrategy
{
  public function apply(...)//сигнатура как в ScenarioTester::test()
  {
    $testResults = [];
    foreach($this->versionCollection as $version) {
      $isolatedEnvironment = $this->envManager->isolateEnvironment(...);
      $context = $isolatedEnvironment->getContext(VersionedContext::class);
      $context->setVersion($version);//теперь можем подставлять в placeholder #v#

      $setup = $this->decoratedTester->setUp(...);
      $localSkip = $skip || !$setup->isSuccessful();

      $result = $thiss->decoratedTester->test(...);
      $tearDown = $this->decoratedTester->tearDown(...);
      $testResults[] = new TestWithSetupResult($setup, $result, $tearDown);
    }
    return new TestResults($testResults);
  }
}
```





Стратегия тестирования

```
class VersionStrategy
{
  public function apply(...)
  {
    $testResults = [];
    foreach($this->versionCollection as $version) {
     $isolatedEnvironment = $this->envManager->isolateEnvironment(...);
     $context = $isolatedEnvironment->getContext(VersionedContext::class);
     $context->setVersion($version);//теперь можем подставлять в placeholder #v#

     $setup = $this->decoratedTester->setUp(...);
     $localSkip = $skip || !$setup->isSuccessful();

     $result = $thiss->decoratedTester->test(...);
     $tearDown = $this->decoratedTester->tearDown(...);
     $testResults[] = new TestWithSetupResult($setup, $result, $tearDown);
    }
    return new TestResults($testResults);
  }
}
```





Стратегия тестирования

```
class VersionStrategy
{
  public function apply(...)
  {
    $testResults = [];
    foreach($this->versionCollection as $version) {
     $isolatedEnvironment = $this->envManager->isolateEnvironment(...);
     $context = $isolatedEnvironment->getContext(VersionedContext::class);
     $context->setVersion($version);//теперь можем подставлять в placeholder #v#

     $setup = $this->decoratedTester->setUp(...);
     $localSkip = $skip || !$setup->isSuccessful();

     $result = $thiss->decoratedTester->test(...);
     $tearDown = $this->decoratedTester->tearDown(...);
     $testResults[] = new TestWithSetupResult($setup, $result, $tearDown);
    }
    return new TestResults($testResults);
  }
}
```





Стратегия тестирования

```
class VersionStrategy
{
  public function apply(...)
  {
    $testResults = [];
    foreach($this->versionCollection as $version) {
     $isolatedEnvironment = $this->envManager->isolateEnvironment(...);
     $context = $isolatedEnvironment->getContext(VersionedContext::class);
     $context->setVersion($version);//теперь можем подставлять в placeholder #v#

     $setup = $this->decoratedTester->setUp(...);
     $localSkip = $skip || !$setup->isSuccessful();

     $result = $thiss->decoratedTester->test(...);
     $tearDown = $this->decoratedTester->tearDown(...);
     $testResults[] = new TestWithSetupResult($setup, $result, $tearDown);
    }
    return new TestResults($testResults);
  }
}
```



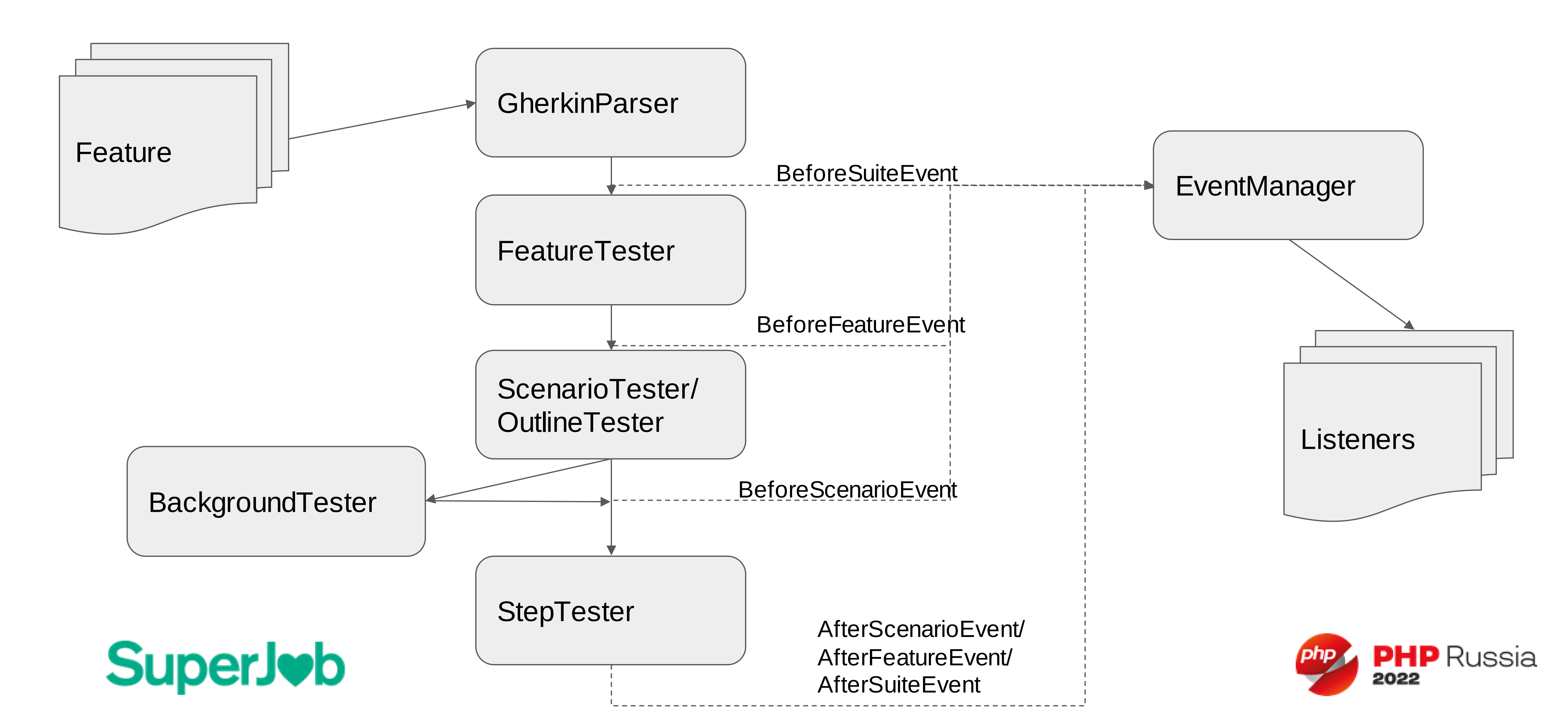


Работаем с Reporter





Работаем с Reporter



Работаем с Reporter


```
▼<testsuites totalTime="19209.942" totalEconomyTime="9832.675" economyPercent="33.9%">
 ▼<testsuite name="HR-бот: статистика по рассылкам HR-бота" tests="55" assertions="0" failures="0" errors="0" time="1716.799" rerun="0" compacted="18"
  economyTime="842.224" economyPercent="32.9%">
    <testcase name="/upcoming/... получаем статистику по всем рассылкам клиента" rerun="false" time="47.143" status="passed" version="upcoming"/>
    <testcase name="/20.0/... получаем статистику по всем рассылкам клиента" rerun="false" time="0" status="passed" version="20.0" wasCompacted="upcoming"
    referenceTime="47.143"/>
    <testcase name="/19.0/... получаем статистику по всем рассылкам клиента" rerun="false" time="45.466" status="passed" version="19.0"/>
    <testcase name="/18.3/... получаем статистику по всем рассылкам клиента" rerun="false" time="45.719" status="passed" version="18.3"/>
    <testcase name="/17.0/... получаем статистику по всем рассылкам клиента" rerun="false" time="45.649" status="passed" version="17.0"/>
    <testcase name="/16.1/... получаем статистику по всем рассылкам клиента" rerun="false" time="50.436" status="passed" version="16.1"/>
    <testcase name="/15.0/... получаем статистику по всем рассылкам клиента" rerun="false" time="0.001" status="passed" version="15.0" wasCompacted="16.1"
    referenceTime="50.436"/>
    <testcase name="/14.2/... получаем статистику по всем рассылкам клиента" rerun="false" time="0.001" status="passed" version="14.2" wasCompacted="16.1"
    referenceTime="50.436"/>
    <testcase name="/13.0/... получаем статистику по всем рассылкам клиента" rerun="false" time="45.548" status="passed" version="13.0"/>
    <testcase name="/12.2/... получаем статистику по всем рассылкам клиента" rerun="false" time="0.001" status="passed" version="12.2" wasCompacted="13.0"
    referenceTime="45.548"/>
    <testcase name="/11.0/... получаем статистику по всем рассылкам клиента" rerun="false" time="46.292" status="passed" version="11.0"/>
```






Выпуск новой версии, тэги неопределенности





Выпуск новой версии, тэги неопределенности

```
Feature: Users

@version-to-???
Scenario: Изменение имени
юзера
  Given …

@version-from-???
Scenario: Изменение имени
юзера
  Given …
```

Release ???

(last: ???)





Выпуск новой версии, тэги неопределенности

```
Feature: Users

@version-to-latest
Scenario: Изменение имени
юзера
  Given …

@version-from-upcoming
Scenario: Изменение имени
юзера
  Given …
```

Release ???

(last: ???)





Выпуск новой версии, тэги неопределенности

```
Feature: Users

@version-to-latest
Scenario: Изменение имени юзера
  Given …

@version-from-upcoming
Scenario: Изменение имени юзера
  Given …
```

Release 9.0

(last: 8.7)

```
Feature: Users

@version-to-8.7
Scenario: Изменение имени юзера
  Given …

@version-from-9.0
Scenario: Изменение имени юзера
  Given …
```





Ускоряем тестирование

Время работы сценариев для всех версий

18 версий





Время работы сценариев для всех версий

18 версий * 3 293 сценария





Время работы сценариев для всех версий

18 версий * 3 293 сценария = 59 274 теста





Время работы сценариев для всех версий

18 версий * 3 293 сценария = 59 274 теста * 0,81 секунды





Время работы сценариев для всех версий

18 версий * 3 293 сценария = 59 274 теста * 0,81 секунды ~ 13 часов





Время работы сценариев для всех версий

18 версий * 3 293 сценария = 59 274 теста * 0,81 секунды ~ 13 часов

52 996 тестов по всем версиям

Время прохождения — 12 часов (~1,5 часа)





Параллельный запуск тестов

Параллельный запуск тестов*

Liuggio\Fastest

```
behat.yml
...
extensions:
    Liuggio\Fastest\Behat2\ListFeaturesExtension\Extension: ~
```

```
$  php /my/path/behat --list-features | php vendor/bin/fastest --process=8 "/my/path/behat {}"
```





Параллельный запуск тестов*

Liuggio\Fastest

```
behat.yml
...
extensions:
    Liuggio\Fastest\Behat2\ListFeaturesExtension\Extension: ~
```

```
$  php /my/path/behat --list-features | php vendor/bin/fastest --process=8 "/my/path/behat {}"
```

* Можно оптимизировать, если отдавать список фич отсортированным





Время работы сценариев для всех версий

18 версий * 3 293 сценария = 59 274 теста * 0,81 секунды ~ 13 часов

52 996 тестов по всем версиям

Время прохождения — 12 часов (~1,5 часа)





Флаги deprecated

DEPRECATED





Фильтруем deprecated версии

```
class VersionTestFilterFactory
{
    private array $excludeVersionsFilters = [];

  public function __construct(..., $skipDeprecatedVesions = false)
    {
        ...
        if($skipDeprecatedVesions) {
            $skippedVersions = this->versionHistory->getDeprecatedVersions();
            $this->excludeVersionsFilters[] = new ExcludeVersionsFilter($skippedVersions);
        }
        ...
     }
}
```





Время работы сценариев для всех версий

18 версий * 3 293 сценария = 59 274 теста * 0,81 секунды ~ 13 часов

52 996 тестов по всем версиям

Время прохождения — 12 часов (~1,5 часа)

32 528 тестов non deprecated

Время прохождения — 7,5 часов (~55 минут)





Compact сценариев

НЕТ ВРЕМЕНИ ОБЪЯСНЯТЬ





Impact-анализ в тестировании

Сценарий “Test 1”

API

Backend

⬠	◔	✚	GET A	GET C	1.0
⬠	◔	✚	GET A	GET C	2.0
⬠	☼	✚	GET A’	GET C	3.0

Тестирование версий

Сценарий 1	Сценарий 2	Сценарий 3	Сценарий 4	Сценарий 5	1.0
Сценарий 1	Сценарий 2	Сценарий 3	Сценарий 4	Сценарий 5	2.0
Сценарий 1	Сценарий 2	Сценарий 3	Сценарий 4	Сценарий 5	3.0
Сценарий 1	Сценарий 2	Сценарий 3	Сценарий 4	Сценарий 5	4.0





Тестирование версий

Сценарий 1	Сценарий 3				1.0
Сценарий 1	Сценарий 3	Сценарий 4			2.0
Сценарий 1	Сценарий 3	Сценарий 4	Сценарий 5		3.0
Сценарий 1	Сценарий 2	Сценарий 3	Сценарий 4	Сценарий 5	4.0





Схлопываем версии

```
class VersionStrategy
{
  public function apply(...)
  {
   $testResults = [];
   foreach($this->versionCollection as $version) {
     $isolatedEnvironment = $this->envManager->isolateEnvironment(...);
     $context = $isolatedEnvironment->getContext(VersionedContext::class);
     $context->setVersion($version);//теперь можем подставлять в placeholder #v#

     $setup = $this->decoratedTester->setUp(...);
     $localSkip = $skip || !$setup->isSuccessful();

     $result = $thiss->decoratedTester->test(...);
     $tearDown = $this->decoratedTester->tearDown(...);
     $testResults[] = new TestWithSetupResult($setup, $result, $tearDown);v
   }
   return new TestResults($testResults);
  }
}
```





Схлопываем версии

```
class VersionStrategy
{
  public function apply(...)
  {
    $testResults = [];
    foreach($this->versionCollection->getReverseIterator() as $version) {
      if ($this->compactScenarios
        && $this->isEqualReferenceEndpointProcessInfoForVersion($versionString)
      ) {
        $testResults[] = $this->compactScenarioInsteadRun($feature, $versionScenario, $skip);
      } else {
        $this->resetReferenceEndpointProcessInfo();
        $testResults[] = $this->run($environment, $feature, $versionScenario, $skip);
      }
    }
    return new TestResults($testResults);
  }
}
```





Схлопываем версии

```
class VersionStrategy
{
  public function apply(...)
  {
   $testResults = [];
   foreach($this->versionCollection->getReverseIterator() as $version) {
    if ($this->compactScenarios
     && $this->isEqualReferenceEndpointProcessInfoForVersion($versionString)
    ) {
     $testResults[] = $this->compactScenarioInsteadRun($feature, $versionScenario, $skip);
    } else {
     $this->resetReferenceEndpointProcessInfo();
     $testResults[] = $this->run($environment, $feature, $versionScenario, $skip);
    }
   }
   return new TestResults($testResults);
  }
}
```





Схлопываем версии

```
class VersionStrategy
{
  public function apply(...)
  {
    $testResults = [];
    foreach($this->versionCollection->getReverseIterator() as $version) {
      if ($this->compactScenarios
        && $this->isEqualReferenceEndpointProcessInfoForVersion($versionString)
      ) {
        $testResults[] = $this->compactScenarioInsteadRun($feature, $versionScenario, $skip);
      } else {
        $this->resetReferenceEndpointProcessInfo();
        $testResults[] = $this->run($environment, $feature, $versionScenario, $skip);
      }
    }
    return new TestResults($testResults);
  }
}
```





Время работы сценариев для всех версий

18 версий * 3 293 сценария = 59 274 теста * 0,81 секунды ~ 13 часов

52 996 тестов по всем версиям

Время прохождения — 12 часов (~1,5 часа)

32 528 тестов non deprecated

Время прохождения — 7,5 часов (~55 минут)

35 541 тестов по всем версиям - compacted

Время прохождения — 8 часов (~1 час)

21 349 тестов non deprecated - compacted

Время прохождения — 5 часов (~37 минут)





Режимы запуска тестов

Режим работы:
с 8 до 20
перерыв с 13 до 14
Суббота с 9 до 18
Выходной ОСЕНЬ





Как фильтруются версии

```
class VersionTestFilterFactory
{

  public function getFilters(VersionRange $versionRange)
  {
    $filters = [];
    ...
    if(null !== $versionRange->getVersionFrom()) {
     $filters[] = new GreaterOrEqualFilter($versionRange->getVersionFrom());
    }
    if(null !== $versionRange->getVersionTo()) {
     $filters[] = new LessOrEqualFilter($versionRange->getVersionTo());
    }
    ...
    return array_merge($filters, $this->excludeVersionsFilters);
   }
}
```





Режимы запуска тестов

FULL

Тестируем всё, что есть

Режим не добавляет фильтров





Режимы запуска тестов

FULL

Тестируем всё, что есть
Режим не добавляет фильтров

MAJOR

Тестируем только старшие версии в каждой мажорной версии.
фильтр ***OnlyMajorVersion***





Режимы запуска тестов

FULL
Тестируем всё, что есть
Режим не добавляет фильтров

MAJOR
Тестируем только старшие версии в каждой мажорной версии.
фильтр ***OnlyMajorVersion***

STRICT
Тестируем строго одну версию.
В командную строку, помимо режима, передается версия.
фильтр ***EqualFilter***





Режимы запуска тестов

FULL
Тестируем всё, что есть
Режим не добавляет фильтров

MAJOR
Тестируем только старшие версии в каждой мажорной версии.
фильтр ***OnlyMajorVersion***

STRICT
Тестируем строго одну версию.
В командную строку, помимо режима, передается версия.
фильтр ***EqualFilter***

MAX
Тестируем только самую последнюю версию, подходящую для сценария (лучше, чем MAJOR)
фильтр ***MaxVersionFilter***

Время работы сценариев для всех версий

18 версий * 3 293 сценария = 59 274 теста * 0,81 секунды ~ 13 часов

52 996 тестов по всем версиям

Время прохождения — 12 часов (~1,5 часа)

32 528 тестов non deprecated

Время прохождения — 7,5 часов (~55 минут)

35 541 тестов по всем версиям - compacted

Время прохождения — 8 часов (~1 час)

21 349 тестов non deprecated - compacted

Время прохождения — 5 часов (~37 минут)

3 293 тестов в режиме max version

Время прохождения — 46 минут (~6 минут)



Время работы сценариев для всех версий

18 версий * 3 293 сценария = 59 274 теста * 0,81 секунды ~ 13 часов

52 996 тестов по всем версиям

Время прохождения — 12 часов (~1,5 часа)

32 528 тестов non deprecated

Время прохождения — 7,5 часов (~55 минут)

35 541 тестов по всем версиям - compacted

Время прохождения — 8 часов (~1 час)

21 349 тестов non deprecated - compacted

Время прохождения — 5 часов (~37 минут)

3 293 тестов в режиме max version

Время прохождения — 46 минут (~6 минут)



Итоги





Мы узнали

- Что такое функциональное тестирование, Behat и Gherkin
- Как работать с тестами в Behat и как их улучшить
- Как тестировать WebApi, учитывая версии
- Как мы доработали Behat для ускорения прохождения тестов





Известные проблемы

- Не решили проблему с версионированием Background

Известные проблемы

- Не решили проблему с версионированием Background
- Не решили проблему затрат в работе с тестовым API (tapi) по сети





Известные проблемы

- Не решили проблему с версионированием Background
- Не решили проблему затрат в работе с тестовым API (tapi) по сети
- Поддержка scope (API Gateway)





Ссылки



Behat Framework

https://docs.behat.org/

https://github.com/Behat/Behat

Параллельный запуск тестов

https://github.com/liuggio/fastest

Доклад на PhpRussia'21. Версионирование API. Единая кодовая база для всех версий

https://www.youtube.com/watch?v=KccOSxI9WJw

SuperJob

Антон Золотилин

Head of Backend

 @djafar_dragon

 anton.zolotilib@gmail.com

Ссылка на доклад: https://clck.ru/32kZcH



PHP Russia 2022

Оцените доклад

